CUDA 프로그래밍 기법 비교 연구

임선영, 박영호
숙명여자대학교 멀티미디어과학과
e-mail : sunnyihm@sm.ac.kr

A Comparison among Methods using CUDA Programming

Sun-Young Ihm, Young-Ho Park
*Dept. of Multimedia Science, Sookmyung Women's University

요 약

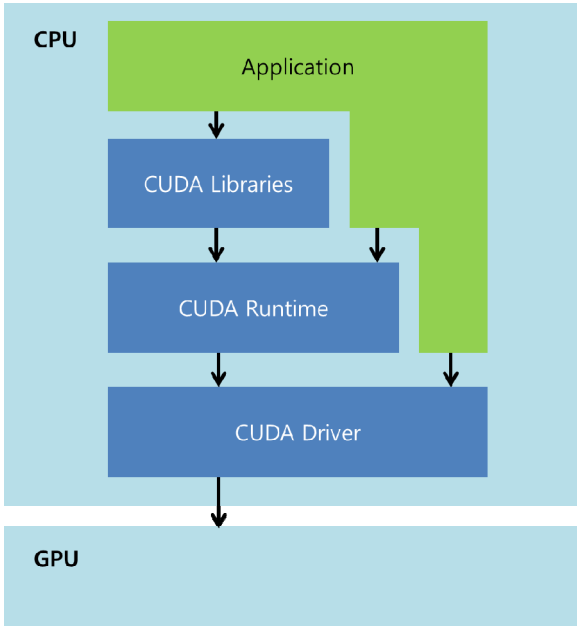
GPU 를 활용하는 병렬 프로그래밍에 대한 관심이 높아지면서 이에 대한 연구가 활발히 진행되고 있다. GPU 의 성능이 높아지면서 이를 일반 연산에 사용하는 방법으로 NVIDIA 사에서 CUDA 프로그래밍 개발 환경을 제공하고 있다. 본 논문에서는 이 CUDA 프로그래밍 기법을 소개하고, 간단한 예제를 통해 CPU 와 GPU 를 사용하는 방법을 비교한다.

1. 서론

병렬 컴퓨팅은 동시에 많은 계산을 하는 연산의 한 방법으로, 최근 GPU(Graphic Processing Unit)의 성능이 높아지면서 GPU 를 활용하는 병렬 프로그래밍에 대한 관심이 높아지고 또 이에 대한 연구가 활발히 진행되고 있다. GPU 는 컴퓨터의 부품 중 하나로 기존에는 그래픽 작업에 사용되었지만, 성능이 향상되면서 CPU 와 같이 범용적으로 사용되고 있다. 이러한 GPU 병렬 프로그래밍을 제공하는 개발 환경의 한 예로 NVIDIA 사에서 제공하는 CUDA(Compute Unified Device Architecture) 개발 환경 [1]을 들 수 있다[2]. CUDA 프로그래밍을 통해 어플리케이션을 제작했을 때, 이전의 구현 방식에 비하여 높은 수치의 성능 향상을 기대할 수 있다. 더욱이, NVIDIA 그래픽스 프로세서로 처리되는 어플리케이션들은 가격대비 우수한 성능 향상이 있으며 전통적인 중앙 처리 기술 전용으로 구현한 것보다도 와트(watt) 대비 우수한 성능 향상을 갖는다[3]. 본 논문에서는 덧셈을 하는 간단한 예제를 통해 CPU 에서와 GPU 에서의 성능을 비교한다. 본 논문의 구성은 다음과 같다. 먼저 2 장에서는 CUDA 프로그래밍 개발 환경을 소개하고, 3 장에서는 비교 분석을 한다. 4 장에서는 결론을 맺는다.

2. CUDA 프로그래밍 개발 환경

CUDA 는 NVIDIA 사에서 개발된, GPU 에서 수행하는 병렬 처리 알고리즘을 C 프로그래밍 언어 등을 사용하여 작성할 수 있도록 하는 기술이다. CUDA 언어는 C 언어를 확장한 것으로 그림 1[1]에서 보는 바와 같이 CUDA 프로그램 개발 환경은 GPU 디바이스 드라이버와 컴파일러 및 런타임 시스템 라이브러리로 이루어져 있다[2].



(그림 1) CUDA 소프트웨어 스택

CUDA 프로그래밍 방법은 그래픽 API 를 사용하는 전통적인 범용 GPU 와 비교하여 몇 가지 장점을 가진다. 먼저, 코드가 메모리의 임의 위치에서 데이터를 읽을 수 있다. 또한, 디바이스 상의 읽기 및 쓰기가 호스트보다 더 빠르다. 마지막으로 정수와 비트 단위 연산을 충분히 지원하며 더욱 빠르다 [4].

3. 프로그램 비교 분석

본 장에서는 CUDA C 를 이용하여 스레드를 이용 하는 것을 덧셈을 하는 간단한 예를 통해 설명한다. 숫자로 구성된 두 개의 리스트가 있으며, 루프를 돌면서 두 리스트의 원소들의 합을 세 번째 리스트에 저장하는 예제이다. 동일한 예제를 CPU 와 GPU 를 사

용하도록 각각 구현하였다. 이 때, CPU 를 사용하는 예제는 멀티코어 CPU 시스템을 대비하여 병렬화하여 구현하였다. GPU 를 사용하는 예제에서는 먼저 GPU 메모리를 할당 한 후, CPU 에서 동일하게 두 개의 리스트를 숫자로 구성한다. 다음으로 두 리스트를 GPU 에 복사한 후, 병렬 처리를 통하여 두 리스트의 각 원소의 합을 세 번째 리스트에 저장한다. 그리고 세 번째 배열을 GPU 에서 CPU 로 복사한 후 결과를 출력한다. 마지막으로 GPU 에 할당된 메모리를 해제한다. CUDA 프로그램은 kernel 함수를 GPU 상에서 수행하도록 하며 이 kernel 함수가 호출될 때 인자를 사용하여 스케쥴 할 스레드의 개수를 정할 수 있다. 이 kernel 함수를 __global__ 이라는 지시자를 사용하여 호출한다. 그림 2 와 그림 3 은 각각 CPU 와 GPU 에서의 덧셈 함수의 코드이다. 이 함수들이 각각의 CPU 및 GPU 에서 호출되어 실행된다. 그림 2 와 그림 3 에서 a 와 b 는 숫자로 구성된 리스트이며 c 는 두 리스트의 덧셈 결과를 저장하는 리스트이다. 변수 n 은 루프의 횟수를 뜻하고, m 은 코어의 개수를 뜻한다.

```
void add(int *a, int *b, int *c)
{
  int id = 0;
  while (id < n)
  {
    c[id] = a[id] + b[id];
    id +=m;
  }
}
```

(그림 2) CPU 에서의 덧셈 함수

```
__global__ void add(int *a, int *b, int *c)
{
  int id = 0;
  if (id < n)
  {
    c[id] = a[id] + b[id];
  }
}
```

(그림 3) GPU 에서의 덧셈 함수

4. 결론

본 논문에서는 GPU 를 사용하여 병렬 처리 프로그래밍을 하는 CUDA 프로그래밍 기법을 소개하고 간단한 덧셈 예제를 통해 CPU 를 사용하는 것과 비교하였다. GPU 를 사용하도록 구현한 프로그램이 CPU 를 사용하는 것보다 루프를 많이 돌릴 경우 성능이 최소 2 배 이상 향상 됨을 알 수 있었다. 향후에 병렬 프로그래밍에 GPU 를 활용한다면 성능상의 효과가 클 것으로 기대된다.

본 연구는 지식경제부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신)의 일환으로 수행하였음. [10041854, 안전한 주거환경을 위한 실시간 위험요소 예측/방지용 스마트 홈 서비스 플랫폼 기술개발]

참고문헌

[1] CUDAToolkit Documents, http://docs.nvidia.com/cuda/index.html

[2] 이화영, 임은진, “CUDA programming environment 을 활용한 Path-Integral Monte Carlo Simulation 의 구현,” 한국산업정보학회 학술대회논문집, 2009.5, pp.196-199.

[3] Jason Sanders, Edward Kandrot, “CUDA BY EXAMPLE: An Introduction to General-Purpose GPU Programming,” Addison-Wesley Professional, 2011.

[4] Wikipedia, http://ko.wikipedia.org/wiki/CUDA